REMARKS

In response to the Office Action mailed on 12th December, 2007, Applicant wishes to enter the following remarks for the Examiner's consideration. Applicant has amended claim 13, 14, 19 and 20 and added new claims 25-29. This response accompanies a Request for Continued Examination (RCE).

Claims 13-29 are pending in the application, claim 1-12 are withdrawn.

In the specification, the paragraph on page 7, lines 1-5 has been further amended as is discussed below.

Rejection of amendment under 35 USC §132(a)

The previously amended paragraph:

> "In general, a loop comprises a prolog, a loop-body and an epilog. The prolog and epilog instructions are generally a subset of the instructions in the ~~body of the~~ loop. The prolog instructions are placed at the start of the loop, before the loop body and are used for priming the pipeline. The epilog instructions are placed at the end of the loop, after the loop body and are used for draining the pipeline. According to the present invention, mechanisms are provided to guard the execution (committing of results) to the correct subset of the loop-body. Consequently, the data path can repeatedly execute the loop-body, thereby eliminating the prolog and epilog code."

has been rejected under 35 USC §132(a). The Applicant respectfully traverses this rejection of the amendment. The amendment is supported in the original specification by, for example, Figure 1 and the description of Figure 1 on page 8, line 4-24, of the specification and by page 6, lines 19-21 of the specification.

In particular, Figure 1 shows prolog instructions at time steps 0-5 that are placed at the start of the loop before the loop body (at time steps 6-8). Figure 1 also shows epilog instructions at time steps 9-14 that are placed after the loop body. When multiple iterations are required, the loop body (at time steps 6-8) is repeated (page 8, lines 17-19). This provides support for the

added sentence: "<u>In general, a loop comprises a prolog, a loop-body and an epilog".</u>
At the start of the computation the pipeline is not filled and the asterisks in time steps 0-3 denote invalid data (page 8, lines 14-15). Thus, the prolog instructions are used to fill (prime) the pipeline. Similarly, the asterisks in the epilog denote invalid data, so the epilog is used to drain the pipeline.
Figure 1 shows a complete loop, including epilog, loop body and prolog.

Page 6, lines 19-21 of the specification read "These deep computational pipelines would require many VLIW control words in the general case both for priming the pipeline (prolog) and draining the pipeline (epilog)." This provides support for the added sentences: "<u>The prolog instructions are placed at the start of the loop, before the loop body and are used for priming the pipeline. The epilog instructions are placed at the end of the loop, after the loop body and are used for draining the pipeline."</u>
The words "body of the" in line 3 of the amended paragraph that were deleted in the previous amendment have been reinstated by the present amendment. A VLIW control word is a compound instruction that may contain individual instructions for multiple functional units. The individual instructions in a control word of prolog or epilog are generally a subset of the individual instructions in the body of the loop.

It is noted that the amended paragraph was incorrectly identified as being on page 6 in the previous response. The amended paragraph is actually and now correctly identified as replacing page 7, lines 1-5 of the original specification. Words inadvertently deleted in the previous amendment have been added back in. In light of the above remarks, Applicant submits that no new matter has been added in the amendment. Application requests that the objection to the amendment be withdrawn.

In a telephone interview with the examiner on January 28, 2008, the examiner agreed that the amendment was supported in the specification.

Rejection of claims under 35 USC §112

Claims 13-24 have been rejected under 35 USC §112 as failing to comply with the written description requirement. Applicant respectfully traverses this rejection of the claims.

As discussed above, Applicant submits that Epilog and Prolog instructions are clearly shown in Figure 1 and are further described on page 8, line 4-24, of the specification, and on page 6, lines 19-20, for example.

Claims 13-24 have also been rejected under 35 USC §112, second paragraph, as being indefinite. Applicant respectfully traverses this rejection of the claims.

Referring again to loop shown Figure 1, the loop has three parts, an epilog, a loop body (which may be repeated) and an epilog. The prolog instructions are at the start of the loop. When the epilog instructions and the prolog instructions are removed, a <u>loop body</u> remains. The loop is still a loop because it has a loop body. The loop is not 'nothing' as the examiner opines, since the loop body (the part that is repeated) has not been removed. Indeed, Figure 2 shows a pipelined program loop having a loop body but no prolog instructions (as in claim 13). In Figure 2, there are no separate prolog instructions at the start of the loop, just a loop body. The loop does not need to be primed using separate prolog instructions. Instead, the prolog is replaced by additional repetitions of the loop body (2 in the example shown in Figure 1) and data validity tags are used. Also in Figure 2, there are no epilog instructions at the end of the loop, just a loop body. The loop does not need to be drained using separate epilog instructions. Instead, the epilog is replaced by additional repetitions of the loop body and data validity tags are used.

Claims 13, 19 and 20 have been amended to clarify the meaning of the terms 'prolog' and 'epilog'.

In a telephone interview with the examiner on January 28, 2008, the examiner agreed that the terms epilog and prolog were properly disclosed in the specification. The Applicant agreed to amend the claims to more clearly define the terms.

In light of the foregoing amendment and remarks, Applicant respectfully submits that the 35 USC §112 rejections of the claims have been overcome. Applicant thus respectfully requests that this basis of rejection of the claims be withdrawn and that a Notice of Allowance for these claims be mailed at the Examiner's earliest convenience.

Rejection of claims under 35 USC §102

Claims 13-16 and 19-24 have been rejected under 35 USC §102(b) as being anticipated by Hennessy and Patterson, "Computer Architecture: A Quantitative Approach", 1996, Morgan Kaufman Publishers, Inc., Second Edition, pages 239-247. Applicant respectfully traverses this rejection of the claims in view of the amendments to the claims.

Applicant submits that the examiner's interpretation of the claim is incorrect and not in accordance with the specification. The terms 'prolog' and 'epilog' are defined in the amended paragraph at the top of page 7 of the specification, for example. Referring to page 240 of the Hennessy reference, the loop is of the form shown in Figure 1 of the application. In this loop, the instruction

```
A[1] = A[1] + B[1];
```

is at the start of the loop and forms the loop prolog. The two instructions

```
B[i+1] = C[i] + D[i];
A[i+1] = A[i+1] +B[i+1];
```

form the body of the loop and are repeated 99 times. The instruction

```
B[101] = C[100] + D[100];
```

is at the end of the loop and forms the epilog.

Claim 13, in contrast, is a method for executing a pipelined program loop having a loop body but *no prolog instructions for priming the pipeline*. One aspect of the present invention is the elimination of separate prolog instructions. Hennessy has not eliminated the prolog instruction.

To eliminate the prolog in the example of page 240, in accordance with one embodiment of the present invention, the complete prolog could be rewritten as

```
        B[1] = C[0] + D[0];
        A[1] = A[1] +B[1];
```

Similarly, to eliminate the epilog, the epilog could be written as

```
        B[101] = C[100] + D[100];
        A[101] = A[101] +B[101];
```

The prolog and epilog can then be incorporated in the loop body, so that the code becomes

```
        for (i=0; i<=100; i=i+1) {
                B[i+1] = C[i] + D[i];
                A[i+1] = A[i+1] +B[i+1];
        }
```

Note that the loop body is now executed for more iterations, but there are no separate prolog or epilog instructions, just a loop body. However, the memory locations C[0] and D[0] do not exist, so the corresponding values would be tagged as invalid (I.e. the inputs to the adder would tagged as invalid), and the result of the addition would therefore be tagged as invalid and not be sunk to memory location B[1]. Also, the element A[101] would be tagged as invalid since the source counter for A would expire at i=99. The result of the addition would tagged as invalid and not be saved.

Hennessy discloses a scoreboard that indicates the status of each functional element. When a functional element has completed an operation, its output is always marked as valid regardless of whether the input data was valid or invalid. However, Hennessy is concerned with dynamic scheduling of instructions, rather than the elimination of prolog code. In particular, if an input is not valid (i.e. has not yet been computed for this current iteration), the functional element does not perform the operation. Rather, it waits until the inputs data become valid and then performs the operation. In contrast, in the present invention, instruction execution continues whether the data is valid or not. Invalid data, will produce invalid results, which ultimately will be discarded at the end of the pipeline. In the example above, the scoreboard of Hennessy would show the value at B[i+1] used in the second instruction of the loop is invalid until the first instruction has been completed, ensuring that the instructions are completed in order. This is unrelated to prolog elimination.

Claim 13 has been amended to clarify this distinction. In particular, claim 13 has been amended to include the element of "executing the iteration whether or not the input data values are valid". In a telephone interview with the examiner on January 28, 2008, the examiner agreed that this element of the claim is not taught by the Hennessy reference. Claim 13 has been further amended to include the element "setting the output data validity tag to indicate that the resulting output data from the functional unit is invalid if any of the input data values to the functional unit is invalid." Hennessy teaches that the resulting output data value (the output data once the operation is complete) is always valid, since the input data is not used until it is valid.

Claims 14-19 depend from claim 13. Although additional arguments could be made for the patentability of each of the claims 14-19, such arguments are believed unnecessary in view of the above discussion. For example, claim 19 introduces a sink iteration counter. It is clear from the simple example shown in FIG. 3 of the application that the sink iteration counter is not equivalent to the loop iteration counter when the prolog and epilog are eliminated. In the example, only 3 values are sunk from the 5 iterations of the loop. In iterations where the data is tagged as invalid, the data is not sunk and the sink counter is not decremented, whereas the loop iteration counter is adjusted at every iteration of the loop. Claim 19 has been amended to clarify this distinction.

In light of the foregoing amendment and remarks, Applicant respectfully submits that the Hennessy reference does not teach, suggest, disclose or otherwise anticipate the recitations of claims 13-19. Applicant thus respectfully requests that this basis of rejection of the claims be withdrawn and that a Notice of Allowance for these claims be mailed at the Examiner's earliest convenience.

Claims 20. Claim 20 has been amended to clarify that the loop has no epilog instructions for draining the pipeline. Claim 20 has been further amended to clarify that the iteration is executed without committing the data value to memory if the sink iteration counter indicates that all data values have been committed to memory.

As described above with reference to claim 13, the Hennessy reference on page 240 shows the

B[101] = C[100] + D[100];

This instruction is at the end of the loop and forms the epilog. In contrast, claim 20 is a method for executing a pipelined program loop having a loop body but *no epilog instructions.* In the discussion of claim 13 above, an example is given of how the present invention could be used to eliminate the epilog instruction (by incorporating it into the loop body and adding an appropriate guard mechanism).

The examiner asserts that the iteration counter ‘i’ is a sink counter, however, applicant respectfully submits that this assertion is incorrect. The iteration counter counts how many times the loop body has been executed. However, a sink may be accessed multiple times in a single loop. Thus, difference sinks may have different counter values which, in turn, differ from the loop counter value. In addition, if data is invalid in an iteration, it is not sunk. Thus the iteration counter may be adjusted when the sink counter is not.

Referring to the example loop shown in Figure 3 of the specification, it can be seen that the loop body is executed 5 times. The sink counter is decremented at time steps 6, 9 and 12. The counter has expired at time step 15, so the output value is not sunk. The loop iteration counter is modified in time steps 0-5, while the sink counter is unchanged. It is clear from this example that the loop iteration counter is not equivalent to the sink counter. This aspect is also shown in Figure 11 of the specification. If data is found to be invalid at decision block 244, the iteration is completed at block 248 and the sink iteration counter is not decremented at block 254.

Claim 20 has been amended to call for executing the iteration without committing the data value to memory if the sink iteration counter indicates that all data values have been committed to memory. This element of the claim is not taught be the Hennessy reference. In the Hennessy reference, execution of the iterations is halted when the iteration counter expires.

Claims 21-24 depend from claim 20. Although additional arguments could be made for the patentability of each of the claims, such arguments are believed unnecessary in view of the above discussion.

Applicant thus respectfully requests that the basis of rejection of the claims 13-24 be withdrawn and that a Notice of Allowance for these claims be mailed at the Examiner's earliest convenience.

New Claims

Claims 25-29 have been added to the application.

The claims relate to a method for reducing the number of instructions in a program for controlling a computational pipeline of a processor. The claims are supported in the specification by, for example, figures 3 and 4 and the description thereof. Figures 3 and 4 show how instructions for priming and draining the loop pipeline are implemented as one or more iterations of the loop body. Initialization of data validity tags is shown in Figure 9. The use of a source counter is shown in Figure 10, and the use of a sink counter is shown in Figure 11. The use data validity tags (bits) is shown in Figure 12.

Applicant thus respectfully requests that a Notice of Allowance for the claim 25-29 be mailed at the Examiner's earliest convenience.

In light of the foregoing amendments and explanations, applicant submits that all rejections of claims 13-24 have been overcome. Allowance of claims 13-24 and new claims 25-29 is therefore respectfully requested at the Examiner's earliest convenience. Although additional arguments could be made for the patentability of each of the claims, such arguments are believed unnecessary in view of the above discussion. The undersigned wishes to make it clear that not making such arguments at this time should not be construed as a concession or admission to any statement in the Office Action.

In a further matter, Applicant notes that the Office has not updated its file to reflect the amendment to the title of the application, made by Applicant in the previous filing. It is respectfully requested that the Office update its records to reflect this new title:

Method and Apparatus for Elimination of Prolog and Epilog Instructions in a Vector Processor using Data Validity Tags and Sink Counters

Please contact the undersigned if you have any questions regarding this application.

Respectfully submitted,

/Renee' Michelle Leveque/

Renee' Michelle Leveque

Leveque Intellectual Property Law, P.C.
Reg. No. 36,193
221 East Church Street
Frederick, Maryland 21701
301-668-3073
Attorney for Applicant(s)